03. Operadores y expresiones

Programación - 1º DAM

Luis del Moral Martínez

versión 20.10

Contenidos del tema

1. Operadores y expresiones
2. Operadores del lenguaje C++
3. Conversión de tipos

1. Operadores y expresiones

¿Qué es una expresión?

- Es una **secuencia de operandos** que se combina entre sí mediante **operadores**
- Cada expresión produce un resultado
- Si no hay operadores, el resultado es el propio operando
- **Ejemplos**:
  - n + 15
  - n – 48
  - variable / PI

1. Operadores y expresiones

¿Qué es un operador?

- Representa la **operación** que se llevará a cabo con los distintos operandos
- Existen tres categorías: **unarios**, **binarios** o **ternarios**, según el número de operandos afectados
- También se clasifican según la posición respecto al operando:
  - **prefijo**: se coloca delante
  - **infijo**: se coloca en el interior
  - **sufijo**: se coloca detrás
- **Ejemplos**: +, -, *, !, &…

1. Operadores y expresiones

Propiedades importantes

- Cuando las expresiones tienen varios operandos conviene tener en cuenta:
  - **Precedencia**: orden de evaluación de los operadores
  - **Asociatividad**: determina el orden en el que se asocian operandos del mismo tipo (si no hay paréntesis)
  - **Asociatividad por la derecha (D-I)**: si dos operandos tienen la misma prioridad, empezar por la derecha
  - **Asociatividad por la izquierda (I-D)**: si dos operandos tienen la misma prioridad, empezar por la izquierda
  - **Sin asociatividad**: hay operadores en los que no tiene sentido la asociatividad (ejemplo: **sizeof**)

1. Operadores y expresiones

Prioridad y asociatividad de los operadores

Prioridad	Operadores	Asociatividad
1	:: x -> [] ()	I-D
2	++ -- ~ ! - + & * sizeof	D-I
3	.* ->*	I-D
4	* / %	I-D
5	+ -	I-D
6	<< >>	I-D
7	< <= > >=	I-D
8	== !=	I-D

Prioridad	Operadores	Asociatividad
9	&	I-D
10	^	I-D
11	| |	I-D
12	&&	I-D
13	?: (condicional)	D-I
14	= *= /= %= += -= <<= >>= &= | |= ^=	I-D
15	, (operador coma)	I-D

1. Operadores y expresiones

Evaluando expresiones compuestas

- Las expresiones compuestas tienen dos o más operadores
- Según como se agrupen los operandos a los operadores, tendremos un resultado u otro
- Para calcular la expresión se usa la precedencia
- La precedencia puede ser anulada utilizando paréntesis
- Abre el fichero **03_01_precedencia.cpp**

2. Operadores del lenguaje C++

Operadores de asignación (1)

- Permiten asignar un valor a una variable
- Asignación simple: **variable1 = expresión**
- Asignación compuesta: **variable1 = variable2 = variable3 = expresión**
- **Ejemplos**:
  - a = 5;
  - longitud = 2 * PI * radio;

2. Operadores del lenguaje C++

Operadores de asignación (2)

Operador	Uso abreviado	Uso no abreviado	Explicación
=	a = 5;	No aplica	Asigna 5 a la variable a
*=	a *= 5;	a = a * 5;	Multiplica a por 5 y lo guarda en a
/*	a /* 5;	a = a / 5;	Divide a por 5 y lo guarda en a
%=	a %= 5;	a = a % 5;	Guarda en a el resto de a/5
+=	a += 5;	a = a + 5;	Calcula a +5 y lo guarda en a
-=	a -= 5;	a = a – 5;	Calcula a-5 y lo guarda en a

2. Operadores del lenguaje C++

Operadores aritméticos

- Realizan operaciones aritméticas básicas
- Siguen las reglas algebraicas de jerarquía o prioridad

Operador	Tipo entero	Tipo real	Ejemplo
+	Suma	Suma	variable = 4 + 5;
-	Resta	Resta	variable = 4 – 5;
*	Producto	Producto	variable = 4 * 5;
/	Cociente (entero)	Cociente coma flotante	variable = 4 / 5;
%	Resto de división	División en coma flotante	variable = 4 % 5;

2. Operadores del lenguaje C++

Operadores de incremento y decremento

- Permiten incrementar o decrementar una variable
- Si se emplean como **prefijos** en una asignación, el incremento se realiza **antes**
- Si se emplean como **sufijos** en una asignación, el incremento se realiza **después**

Operador	Explicación	Ejemplo
++	n = n + 1;	++n, n++
--	n = n – 1;	--n, n--

2. Operadores del lenguaje C++

Operadores relacionales

- Comprueban una relación entre dos operandos
- Generalmente se utilizan en **sentencias de selección** (if) o de **iteración** (while, for)
- Se suelen utilizar para evaluar una condición (**1 – verdadero** y **0 – falso**)

Operador	Explicación	Ejemplo
==	Igual a	a == b
!=	Distinto a	a != b
>	Mayor que	a > b
<	Menor que	a < b
>=	Mayor o igual que	a >= b
<=	Menor o igual que	a <= b

2. Operadores del lenguaje C++

Operadores lógicos (1)

- Permiten evaluar **condiciones lógicas en expresiones booleanas (verdaderas o falsas)**
- Los operandos a la izquierda de && y || se evalúan **en primer lugar (evaluación cortocircuito)**
- Si el valor de la izquierda determina el resultado, entonces no se evalúa el segundo

Operador	Operación lógica	Ejemplo
!	Negación	! (x >= y)
&&	AND (y)	m < n && i > j
||	OR (o)	m = 5 || n != 10

2. Operadores del lenguaje C++

Operadores lógicos (2)

- Tablas de **verdad** de los operadores

a	!a (not a)
verdadero (1)	falso (0)
falso (0)	verdadero (1)

a	b	a && b
verdadero (1)	**verdadero (1)**	verdadero (1)
verdadero (1)	**falso (0)**	falso (0)
falso (0)	**verdadero (1)**	falso (0)
falso (0)	**falso (0)**	falso (0)

a	b	a && b
verdadero (1)	**verdadero (1)**	verdadero (1)
verdadero (1)	**falso (0)**	verdadero (1)
falso (0)	**verdadero (1)**	verdadero (1)
falso (0)	**falso (0)**	falso (0)

2. Operadores del lenguaje C++

Operadores de manipulación de bits

- Permiten manipular bits a **bajo nivel**
- Ejecutan operaciones lógicas sobre cada uno de los bits de los operandos
- Las siguientes operaciones se pueden aplicar sobre **patrones de bits (int, char o long)**

Operador	Operación	Ejemplo
&	AND (y)	a & b
!=	OR (o)	a | | b
^	XOR (o exclusivo, a o b, no los 2)	a ^ b
~	Inversión de bits (0->1 y 1->0)	~a
<<	Desplazar bits a la izquierda	a << 3 (desplaza 3 bits)
>>	Desplazar bits a la derecha	a >> 3 (desplaza 3 bits)

2. Operadores del lenguaje C++

Operador condicional

- Este operador es de tipo ternario
- **Devuelve un valor que depende de la condición comprobada**
- Tiene la siguiente forma:
  - **condición ? expresión_verdadero : expresión_falso;**
  - Si la condición es **verdadera** se devuelve expresión_verdadero
  - Si la condición es **falsa** se devuelve expresión_falso;

2. Operadores del lenguaje C++

Otros operadores (1)

- **Operador coma (,)**
  - Permite combinar dos o más expresiones en una misma línea
  - Ejemplos:
    - `int i = 10, j = 12;`
    - `i++, j++;`
- **Operador ( )**: se utiliza para las llamadas a funciones (entre paréntesis van los argumentos)
- **Operador [ ]**: se utiliza para designar los elementos de un array (lo veremos más adelante)

2. Operadores del lenguaje C++

Otros operadores (2)

- **Operador ::** operador de ámbito de resolución
  - Especifica el alcance o ámbito de un objeto
- **Operador sizeof**: permite conocer el tamaño en bytes de un tipo de dato o variable
  - Ejemplo:
    - `sizeof int`
- **Operadores de direcciones (*, &, . y ->)**
  - Se usarán en el tema de punteros y memoria dinámica
- Abre el fichero **03_02_operadores.cpp**

3. Conversión de tipos

Convertir un tipo a otro

- Es posible convertir un tipo de valor a otro
- C++ convierte valores:
  - Cuando se asigna un valor aritmético a otra variable aritmética
  - Cuando se combinan tipos mezclados en expresiones
  - Cuando se pasan argumentos a funciones
- Existen dos tipos de conversiones:
  - **Implícitas**: se ejecutan automáticamente
  - **Explícitas**: son solicitadas expresamente por el programador al compilador

3. Conversión de tipos

Conversión implícita (1)

- Esta conversión se ejecuta automáticamente
- Los operadores de precisión más baja (más pequeña) se convierten a los de más precisión
- Es posible que el compilador nos avise cuando se realizan estas conversiones
- Ejemplo:
  - **double pi = 3.141592 + 3;** (conversión de int a double)

3. Conversión de tipos

Conversión implícita (2)

- Estas conversiones tienen algunas reglas:
  1. En una expresión aritmética, los operadores binarios deben tener operandos del mismo tipo
     - Si no tienen el mismo tipo, uno de los operandos se convertirá al tipo del otro antes de evaluar la expresión
  2. En una llamada a una función, el tipo de argumento debe corresponder con el tipo del parámetro
     - Si no coincide, será convertido para que coincida con su tipo
  3. C++ posee operadores de moldeado de tipos que permiten convertir los tipos

3. Conversión de tipos

Conversión implícita (3)

- El lenguaje C++ define ciertas conversiones aritméticas entre los tipos de datos básicos
- Las más comunes son las conversiones aritméticas
- La conversión permite promocionar los datos, de un tipo más pequeño a un tipo mayor

3. Conversión de tipos

Conversión implícita (4)

- Orden de los tipos de datos
- La siguiente tabla muestra el orden de los tipos de datos frente a una conversión automática:

Tipo de dato	Precisión	Orden
long double	10 bytes	Más alta (19 dígitos)
double	8 bytes	15 dígitos
float	4 bytes	7 dígitos
long	4 bytes	-
int	4 bytes	-
short	2 bytes	-
char	1 byte	más bajo
bool	no aplica	-

3. Conversión de tipos

Conversión explícita (1)

- La conversión explícita permite forzar la conversión de tipos
- Esta conversión es solicitada por el programador al compilador
- La conversión tiene la forma **tipo (expresión)** o **(tipo) expresión;**
- Ejemplos:
  - **(float) i; // convierte i a float**
  - **float (i); // convierte i a float**
- Abre el fichero **03_03_conversion.cpp**

Créditos de las imágenes y figuras

Cliparts e iconos

- **Obtenidos mediante la herramienta web IconFinder** (según sus disposiciones):
  - Diapositiva 1
  - Según la plataforma IconFinder, dicho material puede usarse libremente (free comercial use)
  - A fecha de edición de este material, todos los cliparts son free for comercial use (sin restricciones)

Resto de diagramas y gráficas

- Se han desarrollado en PowerPoint y se han incrustado en esta presentación
- Todos estos materiales se han desarrollado por el autor
  - Si se ha empleado algún icono externo, este se rige según lo expresado anteriormente